Lab 10

SQL Injection Attacks

Estimated time to complete: 40 Min

WHAT YOU WILL LEARN

How to trick websites into giving you the ability to run commands directly on their database, allowing you to pull any information you'd like.

WHY IT'S IMPORTANT

SQL injection is one of the most common web hacking techniques.

An SQL injection attack could destroy your database or could result in data breaches exposing private records and company assets to the public.

SKILLS GAINED

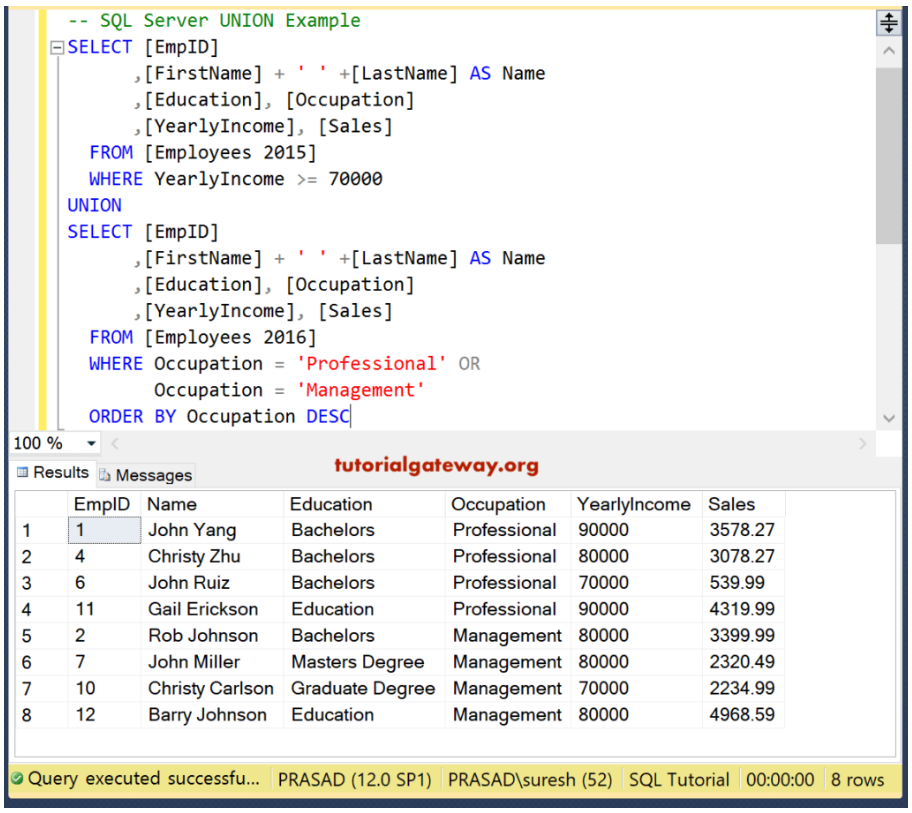
- Database management
- Web penetration testing
- SQL injection

REQUIRED HARDWARE

- Raspberry Pi
- Internet connection

WHAT IS SQL?

SQL, or Structured Query Language, is a way of querying a backend database and pulling out information you want. It is sort of like a programming language, you are defining a string of commands and SQL takes those commands to pull information out of the database. This is the most common way websites organize products, account information, and various other large data sets.



```
-- SQL Server UNION Example
SELECT [EmpID]
      ,[FirstName] + ' ' +[LastName] AS Name
      ,[Education], [Occupation]
      ,[YearlyIncome], [Sales]
  FROM [Employees 2015]
  WHERE YearlyIncome >= 70000
UNION
SELECT [EmpID]
      ,[FirstName] + ' ' +[LastName] AS Name
      ,[Education], [Occupation]
      ,[YearlyIncome], [Sales]
  FROM [Employees 2016]
  WHERE Occupation = 'Professional' OR
        Occupation = 'Management'
  ORDER BY Occupation DESC
```

100 %

Results | Messages

tutorialgateway.org

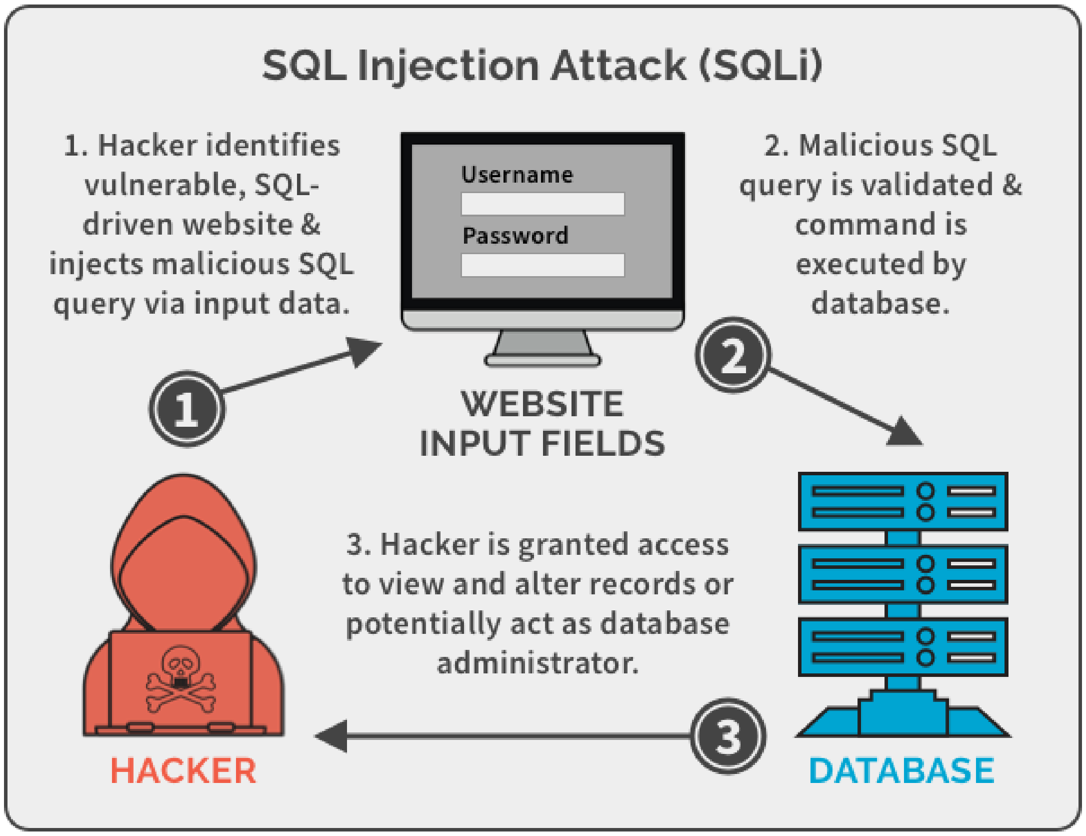
	EmpID	Name	Education	Occupation	YearlyIncome	Sales
1	1	John Yang	Bachelors	Professional	90000	3578.27
2	4	Christy Zhu	Bachelors	Professional	80000	3078.27
3	6	John Ruiz	Bachelors	Professional	70000	539.99
4	11	Gail Erickson	Education	Professional	90000	4319.99
5	2	Rob Johnson	Bachelors	Management	80000	3399.99
6	7	John Miller	Masters Degree	Management	80000	2320.49
7	10	Christy Carlson	Graduate Degree	Management	70000	2234.99
8	12	Barry Johnson	Education	Management	80000	4968.59

```
Query executed successfu... | PRASAD (12.0 SP1) | PRASAD\suresh (52) | SQL Tutorial | 00:00:00 | 8 rows
```

WHAT IS A SQL INJECTION ATTACK?

While we may never see what a SQL command looks like on a website, every time we type something into a search box, that website parses our request into an SQL query statement and sends it to the database for retrieval of our requested information. When this query goes back to the database, part of it is our data (what we searched for) and part of it is the SQL commands needed to interact with the database. This is where the inherent danger of SQL lies.

An SQL database has no idea what parts of this request are supposed to be data or what parts are supposed to be code unless the website purposely separates them—this is where SQL injection comes in. A malicious user could begin typing commands after their data entry and ask the SQL database to do whatever they want, giving them essentially direct unsecured access to the backend database.


SQL Injection Attack (SQLi)
1. Hacker identifies vulnerable, SQL-driven website & injects malicious SQL query via input data.
Username
Password
2. Malicious SQL query is validated & command is executed by database.
1
2
WEBSITE INPUT FIELDS
3. Hacker is granted access to view and alter records or potentially act as database administrator.
3
HACKER
DATABASE

WHY ARE SQL INJECTION ATTACKS IMPORTANT?

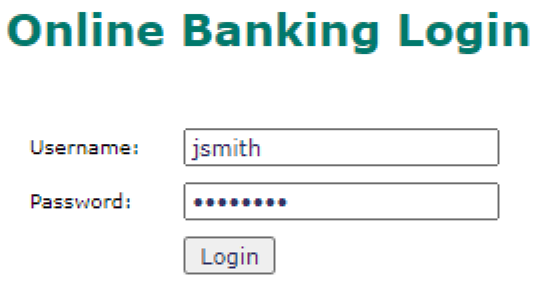
SQL injection attacks are the most common and possibly most damaging web based attack in existence, but cybersecurity researchers say SQLI is one of the least sophisticated, easy-to-defend against cyberthreats. So why does it still happen? Steps need to be taken on a website to "scrub" commands so that attackers cannot add on SQL statements to an input box.

HOW TO PERFORM AN SQL INJECTION ATTACK

DISCLAIMER: THE WEBSITE WE ARE USING IS DESIGNED WITH THE PURPOSE OF TESTING WEB APPLICATION VULNERABILITIES AND WEBSITE DEFECTS. DO NOT ATTEMPT TO PERFORM AN INJECTION ATTACK ON OTHER WEBSITES UNLESS THEY ARE SPECIFICALLY DESIGNED FOR YOU TO DO SO.

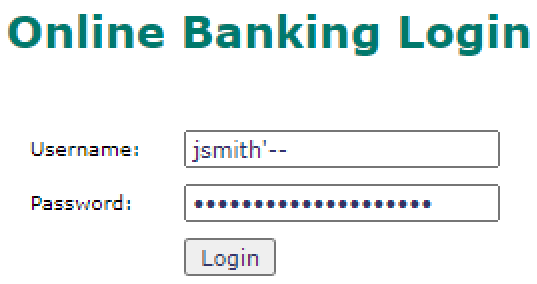
STEP 1: TESTFIRE DEMO

Open a web browser and navigate to ***demo.testfire.net***, this is a demo site created for cybersecurity professionals to test their knowledge and tools. This website is a fake bank with a database on the back end containing accounts, user information, and transactions. For testing purposes, let's say you already know a login at this bank, the username is ***jsmith*** and the password is ***demo1234***. Login with this account at http://demo.testfire.net/login.jsp and browse around the contents of the website, look for valuable information such as the amount of accounts, balances in the accounts and the overall layout of the website.



STEP 1: INJECTION ATTACKS

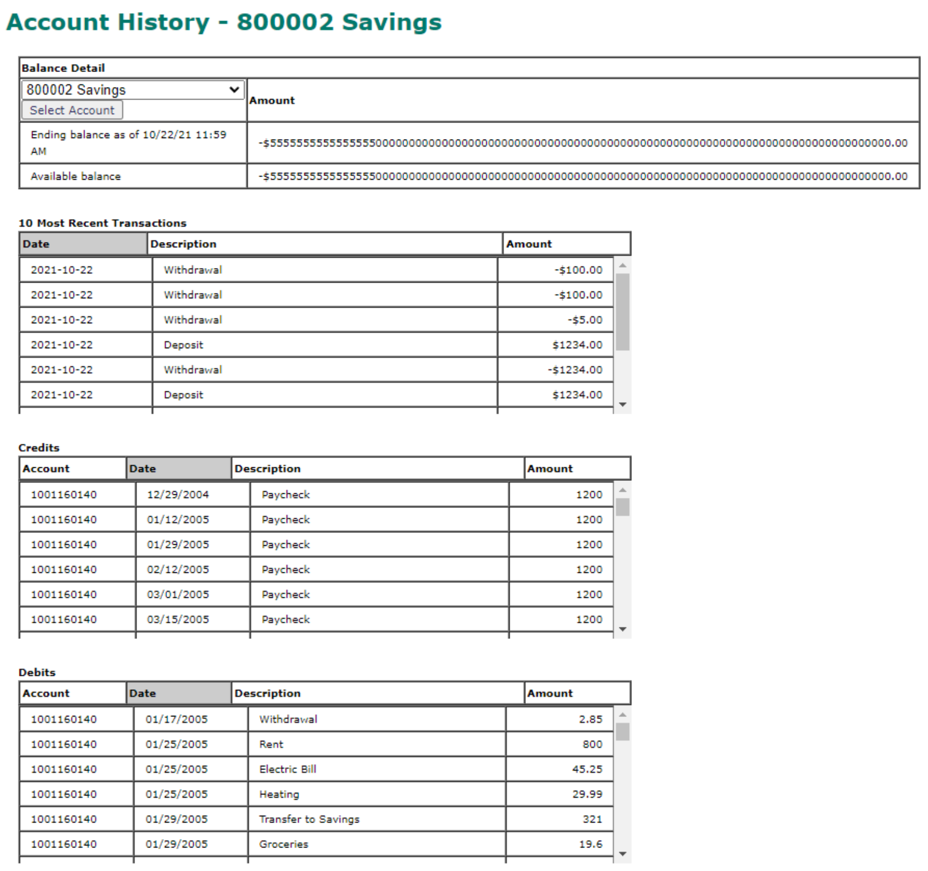
Now, let's say you only know the username but don't know the password. In the username box enter ***jsmith'--*** and then enter any string of random characters into the password field.



Online Banking Login

Username:	jsmith'--
Password:	••••••••••••••••••••

Click login, you should be greeted with the same account page as before, bypassing the password on the account. Click around the accounts to confirm that they are the same as before, telling us that we did in fact log into jsmiths account.



Account History - 800002 Savings

Balance Detail	
800002 Savings	Amount
Ending balance as of 10/22/21 11:59 AM	-$5555555555555555000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000.00
Available balance	-$5555555555555555000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000.00

10 Most Recent Transactions

Date	Description	Amount
2021-10-22	Withdrawal	-$100.00
2021-10-22	Withdrawal	-$100.00
2021-10-22	Withdrawal	-$5.00
2021-10-22	Deposit	$1234.00
2021-10-22	Withdrawal	-$1234.00
2021-10-22	Deposit	$1234.00

Credits

Account	Date	Description	Amount
1001160140	12/29/2004	Paycheck	1200
1001160140	01/12/2005	Paycheck	1200
1001160140	01/29/2005	Paycheck	1200
1001160140	02/12/2005	Paycheck	1200
1001160140	03/01/2005	Paycheck	1200
1001160140	03/15/2005	Paycheck	1200

Debits

Account	Date	Description	Amount
1001160140	01/17/2005	Withdrawal	2.85
1001160140	01/25/2005	Rent	800
1001160140	01/25/2005	Electric Bill	45.25
1001160140	01/25/2005	Heating	29.99
1001160140	01/29/2005	Transfer to Savings	321
1001160140	01/29/2005	Groceries	19.6

This works because the **‘** basically tells the SQL database that this is where our input ends, the **--** characters act as a comment in SQL, so it basically comments out the next SQL statement, which is requesting a password. To the database, a normal login would look like this:

SELECT * FROM users WHERE uid = ‘jsmith’ AND password = md5(‘demo1234’);

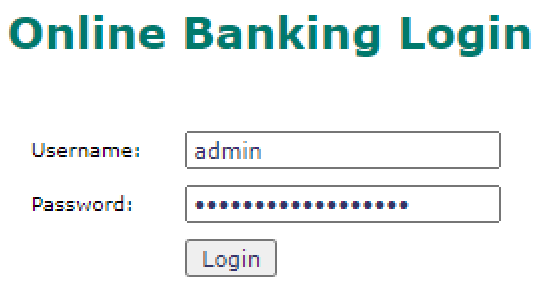
And our request would look something like this:

SELECT * FROM users WHERE uid = ‘jsmith’--’ AND password = md5(‘1234’);

The red part of the statement is now considered by the database to be a comment, which it essentially ignores.

STEP 2: ALTERNATE METHOD

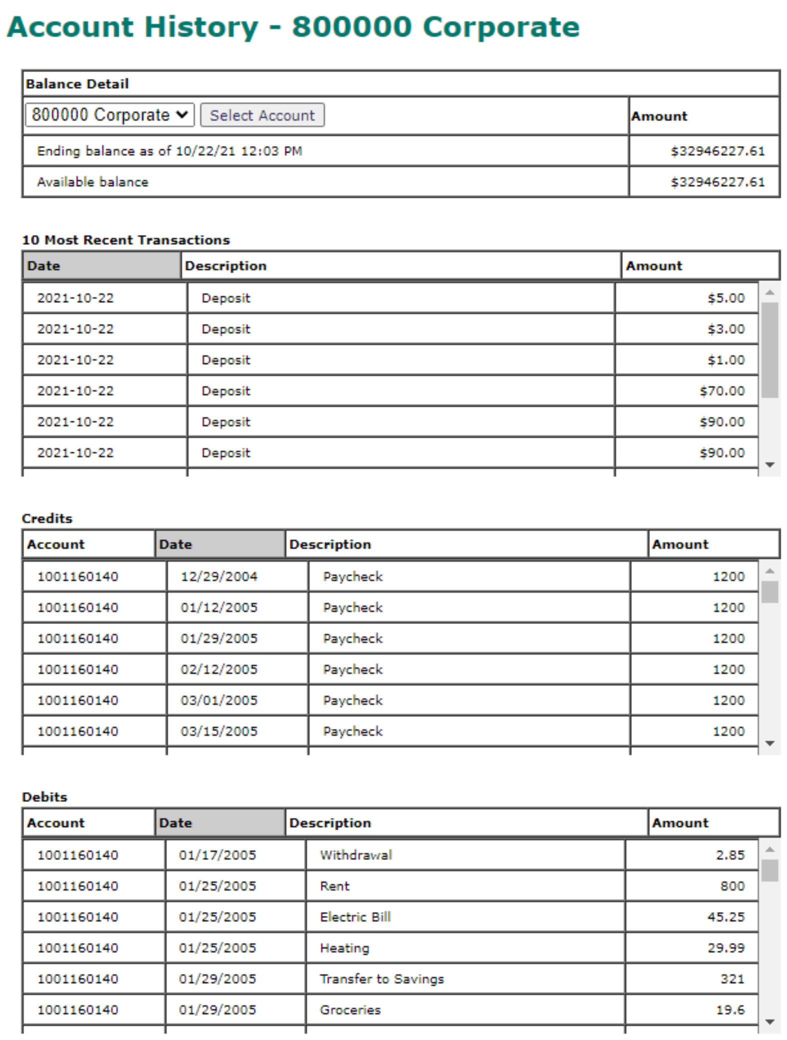
Now that we know this website is vulnerable to an injection attack, lets try to access an admin account. We could use the last method to access the admin account, but let's try another SQL vulnerability to break into this one. We can guess that the admin username is just ***admin,*** so enter that into the username, into the password box enter the phrase: ***unknown’ or ‘1’=’1*** (Unknown can be any random string of characters).


Online Banking Login

Username: admin

Password: ••••••••••••••••••

You should be able to log into the account, click into the accounts to verify that we are logged into an account separate from jsmith’s.



Account History - 800000 Corporate

Balance Detail	
800000 Corporate / Select Account	Amount
Ending balance as of 10/22/21 12:03 PM	$32946227.61
Available balance	$32946227.61

10 Most Recent Transactions

Date	Description	Amount
2021-10-22	Deposit	$5.00
2021-10-22	Deposit	$3.00
2021-10-22	Deposit	$1.00
2021-10-22	Deposit	$70.00
2021-10-22	Deposit	$90.00
2021-10-22	Deposit	$90.00

Credits

Account	Date	Description	Amount
1001160140	12/29/2004	Paycheck	1200
1001160140	01/12/2005	Paycheck	1200
1001160140	01/29/2005	Paycheck	1200
1001160140	02/12/2005	Paycheck	1200
1001160140	03/01/2005	Paycheck	1200
1001160140	03/15/2005	Paycheck	1200

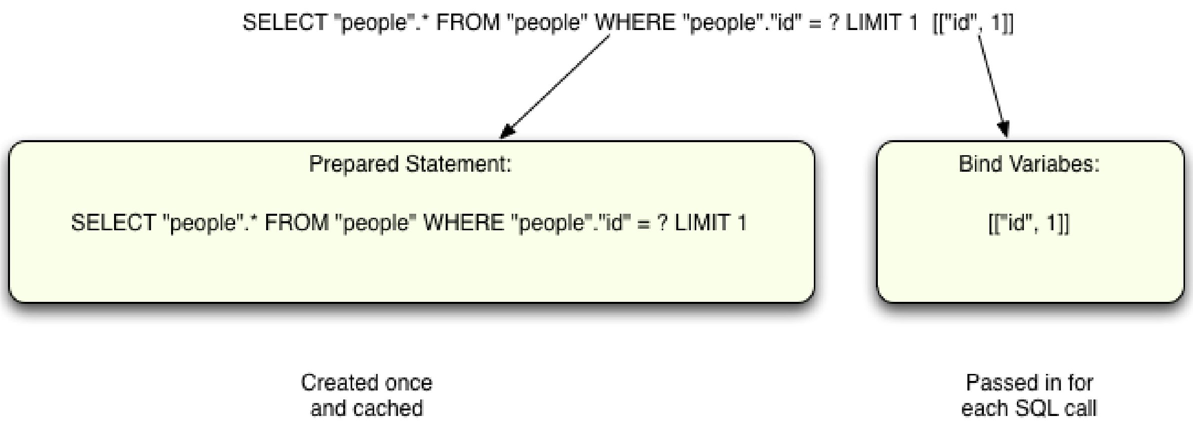
Debits

Account	Date	Description	Amount
1001160140	01/17/2005	Withdrawal	2.85
1001160140	01/25/2005	Rent	800
1001160140	01/25/2005	Electric Bill	45.25
1001160140	01/25/2005	Heating	29.99
1001160140	01/29/2005	Transfer to Savings	321
1001160140	01/29/2005	Groceries	19.6

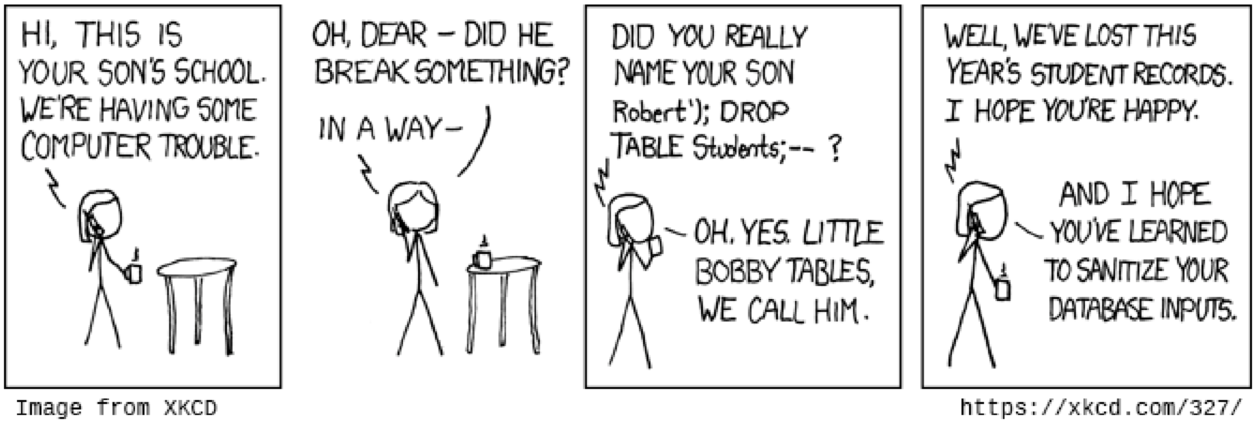
This breaks SQL in a different way. SQL does have some logic built into it, so you end your password input again with the ' character. Even though this is not the password, the string ***or '1'='1*** is read in its place. SQL just reads this string and returns ***true*** to allow you into the account. While the incorrectly entered password will return ***false***, the ***or '1'='1*** part of the string will always return ***true*** since 1 will always equal 1, letting you into the account.

HOW TO DEFEND AGAINST SQL INJECTION ATTACKS

There are a few different prevention mechanisms you can implement to be sure that your site won't fall victim to an SQL injection attack. One of these methods is using Parameterized Statements, sometimes known as Prepared Statements. This method forces you to define the SQL code first, then pass each parameter to the query afterward. By doing this, the database will be able to distinguish between what input is code and what input is data, even if an attacker attempts to inject code into the input box. You should use these whenever you can, as they are the best way to defend against SQL injection.


SELECT "people".* FROM "people" WHERE "people"."id" = ? LIMIT 1 [["id", 1]]
Prepared Statement:
SELECT "people".* FROM "people" WHERE "people"."id" = ? LIMIT 1
Bind Variabes:
[["id", 1]]
Created once and cached
Passed in for each SQL call

Another method used to prevent injection attacks is Sanitizing Inputs. This involves removing any characters that could be used in an injection attack from the input box. This way, when attackers attempt to use a special character to perform an injection attack, their input will be invalid.


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH, YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.
Image from XKCD
https://xkcd.com/327/

EXERCISES:

1: Find user information

A hacker by the name of Steve Speed has taken all of our money! We want to login to his account and get it back. We found out that his username is sspeed. Use SQL injection to log into his account.

2: Create your own SQL statement:

We recently found a list of all the SQL statements that would allow us to hack into the website. One of the commands even lets us login to the default account! However, it is incomplete.

“' OR X--”

Replace the ***X*** with a Boolean data type and use it in the username box. If done correctly, you should be able to put anything in the password box and login to the default user account! What is this account?

REVIEW

1: What is SQL?

A) A hacking technique for stealing user info

B) Coding language to create websites

C) Language for dealing with data in a database

D) A program that looks for vulnerable websites

2: What is the best method to defend against SQL injection attacks?

A) Sanitizing Inputs

B) Parameterized Statements

C) Escaping Inputs

D) Setting up a firewall

3: SQL Injection attacks are the most common yet one of the least sophisticated attacks. (True/False)

4: What can the ‘ character do in an SQLI attack?

A) Start a comment

B) End our current input

C) Nothing

D) Starts a new command

5: Question on sqlmap

A) commands nmap

B) man nmap

C) nmap -c

D) nmap ?

6: sqlmap question

A) To take down a network

B) To steal user information

C) To find vulnerabilities in a network

D) To make sure a network is functioning

Answers: 1:C 2:B 3:T 4:B 5:B 6:C